Science of Computer Programming 7 (1986) 87-97
North-Holland

SMALL PROGRAMMING EXERCISES 12

M. REM

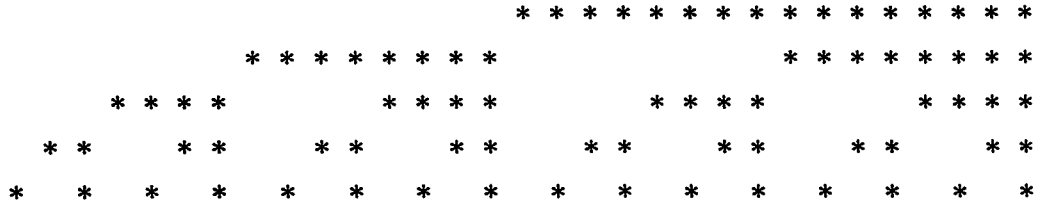
Department of Mathematics and Computing Science, Eindhoven University of Technology, 5600 MB Eindhoven, Netherlands

The country of Alphanumerica has a rather peculiar flag. The first exercise deals with the problem of printing this flag, and in particular with the consequences of iterative versus recursive printing procedures. I owe this problem, including its amusing setting, to J. Nievergelt. He subtitled it "An algorithmic novel about iteration versus recursion". Prof. Nievergelt, now at the University of North Carolina, used this problem in an examination at the ETH in Zurich in the spring of 1985.

The second exercise is another segment problem. A segment of an integer array is called balanced if there occur as many negative as positive values in it. For a given array we have to determine the maximum length of a balanced segment. By introducing an auxiliary array one can obtain a linear solution. This exercise is due to W.H.J. Feijen.

Exercise 30: The flag of Alphanumerica

In the course of a drive to automate her flag industry, the United States of Alphanumerica announced a competition for the most elegant program to print the nation's flag. This rather oblong flag consists of N rows. Row j $(0 \leq j < N)$ is a sequence of 2^j groups, each containing 2^{N-j-1} blanks followed by the same number of stars:



All solutions submitted fell into two categories: the iterative and the recursive ones. The selection of the winning solution sparked a civil war between the adherents of these two algorithmic principles. It split the nation into two: the Iterative States of Alphanumerica, ISA, and the Recursive States of Alphanumerica, RSA. Both nations fly the same flag, but use entirely different manufacturing techniques.

The flag has to be printed by a terminal-like device. For the iterative version the functional specification is

$$
\begin{array}{l}
|[N\colon \mathit{int};\ \{N \geqslant 0\} \\
\quad |[c\colon \mathit{cursor};\ \{c.x = 0 \wedge c.y = 0\} \\
\quad\quad \mathit{fl}(i, j\colon 0 \leqslant i < 2^N \wedge 0 \leqslant j < N)\colon \textbf{array of } (\mathit{blank}, \mathit{star}); \\
\quad\quad \mathit{IS} \\
\quad\quad \{\mathit{fl}(i, j\colon 0 \leqslant i < 2^N \wedge 0 \leqslant j < N) \text{ contains the flag of Alphanumerica}\} \\
\quad]| \\
]|
\end{array}
$$

The specification contains a cursor c, which is a pair $(c.x, c.y)$ of integers. The only way *IS* is allowed to operate on c and *fl* is by the following commands, which have their effects as specified:

$$
\begin{array}{ll}
\mathit{bl}\colon & \mathit{fl}\colon (c.x, c.y) = \mathit{blank};\ c.x := c.x + 1 \\
\mathit{st}\colon & \mathit{fl}\colon (c.x, c.y) = \mathit{star};\ c.x := c.x + 1 \\
\mathit{nl}\colon & c.x, c.y := 0, c.y + 1
\end{array}
$$

The recursive version *RS* involves recursive calls of *RS* that print smaller flags within array *fl*. To express the sizes of these smaller flags we add an integer variable k and allow other initial cursor positions:

$$
\begin{array}{l}
|[N\colon \mathit{int};\ \{N \geqslant 0\} \\
\quad |[k\colon \mathit{int};\ \{k = K \wedge 0 \leqslant K \leqslant N\} \\
\quad\quad c\colon \mathit{cursor};\ \{c.x = X \wedge c.y = Y \wedge 0 \leqslant X \leqslant 2^N - 2^K \wedge 0 \leqslant Y \leqslant N - K\} \\
\quad\quad \mathit{fl}(i, j\colon 0 \leqslant i < 2^N \wedge 0 \leqslant j < N)\colon \textbf{array of } (\mathit{blank}, \mathit{star}); \\
\quad\quad \mathit{RS} \\
\quad\quad \{\mathit{fl}(i, j\colon X \leqslant i < X + 2^K \wedge Y \leqslant j < Y + K) \\
\quad\quad \text{contains the flag of Alphanumerica}\} \\
\quad]| \\
]|
\end{array}
$$

The constants K, X, and Y are, as may be observed in the precondition, the initial values of k, $c.x$, and $c.y$ respectively. Notice that $K = N$ implies $X = 0 \wedge Y = 0$, and that in that case the specification of *RS* coincides with that of *IS*.

The commands for flag printing given earlier do not suffice for recursive solutions. To remedy this we allow in *RS* the use of backspacing and forward and reverse line feed as well:

$$
\begin{array}{ll}
\mathit{back}(E)\colon & c.x := c.x - E \\
\mathit{down}\colon & c.y := c.y + 1 \\
\mathit{up}\colon & c.y := c.y - 1
\end{array}
$$

where $E \geqslant 0$. It seems that the flag industry in RSA needs to be more innovative than its sister industry in ISA. However, an unforeseen consequence of the automation of the flag industry arose: in both countries an increasing number of flags can be seen fluttering in the breeze rotated by 90°. Can the reader explain this?

Exercise 31: Balanced segments

Determine a statement list S such that

$$
\begin{array}{l}
|[\, N\colon \mathit{int};\ \{N \geqslant 0\} \\
\quad X(i\colon 0 \leqslant i < N)\colon \textbf{array of } \mathit{int}; \\
\quad |[\, r\colon \mathit{int}; \\
\qquad S \\
\qquad \{r = (\textbf{MAX}p, q\colon 0 \leqslant p \leqslant q \leqslant N \wedge (\textbf{N}i\colon p \leqslant i < q\colon X(i) < 0) \\
\qquad\qquad\qquad\qquad\qquad\qquad = (\textbf{N}i\colon p \leqslant i < q\colon X(i) > 0)\colon q - p)\} \\
\quad]| \\
]|
\end{array}
$$

Solution of Exercise 27 (A-segments)

With $AS(p, q)$ denoting

$$(\textbf{A}i\colon p \leqslant i \leqslant q\colon X(i) \leqslant abs(X(q)))$$

for $0 \leqslant p \leqslant q \leqslant N$, we have so solve S in

$$
\begin{array}{l}
|[\, N\colon \mathit{int};\ \{N \leqslant 0\} \\
\quad X(i\colon 0 \leqslant i \leqslant N)\colon \textbf{array of } \mathit{int}; \\
\quad |[\, r\colon \mathit{int}; \\
\qquad S \\
\qquad \{r = (\textbf{MAX}p, q\colon 0 \leqslant p \leqslant q \leqslant N \wedge AS(p, q)\colon q - p + 1)\} \\
\quad]| \\
]|
\end{array}
$$

Since

$$AS(p, q) \ \equiv \ AS(p+1, q) \wedge X(p) \leqslant abs(X(q)) \tag{1}$$

for $p < q$, we have

$$AS(p, q) \Rightarrow AS(p+1, q). \tag{2}$$

Such a predicate is called *right-monotonic*. As a consequence, array X should be inspected from right to left. In other words, we obtain the invariant by replacing in the postcondition the lower bound 0 by a variable:

$$
\begin{array}{ll}
P0\colon & 0 \leqslant n \leqslant N \\
& \wedge\ r = (\textbf{MAX}p, q\colon n \leqslant p \leqslant q \leqslant N \wedge AS(p, q)\colon q - p + 1).
\end{array}
$$

When dealing with monotonic predicates we also record the longest 'tail segment', i.e., the longest segment at the boundary n that satisfies AS:

$$P1\colon \qquad s = (\textbf{MAX}q\colon n \leqslant q \leqslant N \wedge AS(n, q)\colon q).$$

From $0 \leqslant n \leqslant N \wedge P1$ we conclude $n \leqslant s$.

Since $AS(N, N)$ holds, invariant $P0 \wedge P1$ may be initialized with $n, r, s = N, 1, N$. We thus arrive at a solution of the following structure:

$$
\begin{array}{l}
n, r, s := N, 1, N\{P0 \wedge P1\} \\
;\textbf{do}\ n \neq 0 \\
\rightarrow\ \{P0 \wedge P1 \wedge 1 \leq n \leq N\}\ S0\ \{P0 \wedge P1^{n}_{n-1} \wedge 1 \leq n \leq N\} \\
;\quad r := r\ \textbf{max}(s - n + 2)\{P0^{n}_{n-1} \wedge P1^{n}_{n-1}\} \\
;\quad n := n - 1\{P0 \wedge P1\} \\
\textbf{od}\ \{P0 \wedge n = N\}
\end{array}
$$

We are left with $S0$, whose purpose it is to establish

$$P1^{n}_{n-1} \quad s = (\textbf{MAX}q\colon n - 1 \leq q \leq N \wedge AS(n-1, q)\colon q).$$

From $P1$ we infer

$$(\textbf{A}j\colon s < j \leq N\colon \neg AS(n, j))$$

which, by (2), implies

$$(\textbf{A}j\colon s < j \leq N\colon \neg AS(n-1, j)).$$

Consequently, $S0$ does not increase s.

By (1) we conclude that if $X(n-1) \leq abs(X(s))$ statement $S0$ is just a **skip**. Next, consider the alternative $X(n-1) > abs(X(s))$. Employing in this case a linear search for

$$(\textbf{MAX}q\colon n - 1 \leq q < s \wedge AS(n-1, q)\colon q)$$

yields a statement $S0$ of the followng structure:

$$
\begin{array}{ll}
S0\colon & \textbf{if}\ X(n-1) \leq abs(X(s)) \rightarrow \textbf{skip} \\
& \square\ X(n-1) > abs(X(s)) \\
& \rightarrow |[q\colon int;\ q := s - 1 \\
& \quad ;\ \textbf{do}\ \neg AS(n-1, q) \rightarrow q := q - 1\ \textbf{od} \\
& \quad ;\ s := q \\
& \quad]| \\
& \textbf{fi}
\end{array}
$$

We rewrite the guard of the repetition. Obviously,

$$X(n-1) > abs(X(q)) \Rightarrow AS(n-1, q)$$

Given $P1 \wedge q < s \wedge X(n-1) > abs(X(s))$, we also have

$$AS(n-1, q) \Rightarrow X(n-1) > abs(X(q))$$

as may be proved as follows:

$$
\begin{array}{ll}
 & AS(n-1, q) \\
\equiv & \{\text{definition of } AS\}
\end{array}
$$

$$
\begin{array}{ll}
 & (\mathbf{E}i\colon n-1 \leq i \leq q\colon X(i) > abs(X(q))) \\
\equiv & \{\text{predicate calculus}\} \\
 & X(n-1) > abs(X(q)) \\
 & \vee\, (X(n-1) \leq abs(X(q)) \wedge (\mathbf{E}i\colon n \leq i \leq q\colon X(i) > abs(X(q)))) \\
\Rightarrow & \{q < s\} \\
 & X(n-1) > abs(X(q)) \\
 & \vee\, (X(n-1) \leq abs(X(q)) \wedge (\mathbf{E}i\colon n \leq i \leq s\colon X(i) > abs(X(q)))) \\
\Rightarrow & \{X(n-1) > abs(X(s))\} \\
 & X(n-1) > abs(X(q)) \vee (\mathbf{E}i\colon n \leq i \leq s\colon X(i) > abs(X(s))) \\
\equiv & \{\text{definition of } AS\} \\
 & X(n-1) > abs(X(q)) \vee \neg AS(n, s) \\
\equiv & \{P1 \Rightarrow AS(n, s)\} \\
 & X(n-1) > abs(X(q))
\end{array}
$$

We may, consequently, replace the guard of the repetition in $S0$ by $X(n-1) > abs(X(q))$, which yields, eliminating variable q,

$S0$:
if $X(n-1) \leq abs(X(s)) \rightarrow$ **skip**
$\square$ $X(n-1) > abs(X(s))$
$\rightarrow s := s-1$
; **do** $X(n-1) > abs(X(s)) \rightarrow s := s-1$ **od**
fi

The above is exactly the first unfolding of

do $X(n-1) > abs(X(s)) \rightarrow s := s-1$ **od**

Thus, we arrive at the following rather compact solution.

S:
$|[n, s\colon int;\ n, r, s := N, 1, N$
; **do** $n \neq 0$
$\rightarrow$ **do** $X(n-1) > abs(X(s)) \rightarrow s := s-1$ **od**
; $r := r$ **max** $(s-n+2)$
; $n := n-1$
od
$]|$

Since the inner repetition makes, during the whole computation, at most N steps, the execution time of S is proportional to N.

Solution of Exercise 28 (length of a longest common subsequence)

We have to determine S such that

$|[M, N\colon int; \{0 \leq M \leq N\}$
$X(i\colon 0 \leq i < M),\ Y(j\colon 0 \leq j < N)$: **array of** *int*;
$|[r\colon int;$

```
    S
    {r = l(M, N)}
  ]|
]|
```

where $l(m, n)$ denotes, for $0 \leq m \leq M$ and $0 \leq n \leq N$, the maximal length of a common subsequence of $X(i: 0 \leq i < m)$ and $Y(j: 0 \leq j < n)$.

Function l has the following recurrence relation:

- for $0 \leq m \leq M$,

$$l(m, 0) = 0,$$

- for $0 \leq n \leq N$,

$$l(0, n) = 0,$$

- for $0 \leq m < M \wedge 0 \leq n < N$,

$$l(m+1, n+1) = \begin{cases} 1 + l(m, n) & \text{if } X(m) = Y(n), \\ l(m, n+1) \ \mathbf{max} \ l(m+1, n) & \text{if } X(m) \neq Y(n). \end{cases}$$

We destroy the symmetry in X and Y and introduce, led by $M \leq N$, an array $a(i: 0 \leq i \leq M)$. We maintain as an invariant

$$P: \quad 0 \leq n \leq N \wedge (\mathbf{A}i: 0 \leq i \leq M: a(i) = l(i, n)).$$

Variable n is initialized at 0. Since $P \wedge n = N$ implies $a(M) = l(M, N)$, our guard is $n \neq N$. The program we thus obtain is

```
S:  |[n: int;
     a(i: 0 ≤ i ≤ M): array of int;
     n := 0
    ; |[i: int; i := 0
     ; do i ≠ M+1 → a: (i) = 0; i := i+1 od
     ]|  {P}
    ; do n ≠ N
     → {P ∧ 0 ≤ n < N}  S0  {P^n_{n+1}}
     ;  n := n+1  {P}
     od {P ∧ n = N}
    ; r := a(M)
    ]|
```

where $S0$ still needs to be elaborated.

Statement $S0$ involves a repetition that maintains

$$Q0: \quad 0 \leq m \leq M \wedge 0 \leq n < N \wedge (\mathbf{A}i: 0 \leq i \leq m: a(i) = l(i, n+1)) \wedge (\mathbf{A}i: m < i \leq M: a(i) = l(i, n))$$

Since $l(0, n+1) = l(0, n) = 0$, $P \wedge 0 \leq n < N \wedge m = 0$ implies $Q0$. We may, consequently, initialize $Q0$ by $m := 0$. Moreover, $Q0 \wedge m = M$ implies P^n_{n+1}.

From the recurrence relation for l we derive the following structure for $S0$:

```
m := 0
; do m ≠ M
  → if X(m) = Y(n) → a: (m+1) = 1 + l(m, n)
    □ X(m) ≠ Y(n) → a: (m+1) = l(m, n+1) max l(m+1, n)
    fi
  ; m := m+1
  od
```

According to $Q0$ we have $l(m, n+1) = a(m)$ and $l(m+1, n) = a(m+1)$. The term $l(m, n)$, however, cannot be retrieved from array a. Therefore, we introduce a variable b and extend our invariant with

$Q1$ $\quad b = l(m, n)$

Thus, our solution becomes

```
S:  |[n: int;
    a(i: 0 ≤ i ≤ M): array of int;
    n := 0
    ; |[i: int; i := 0
    ; do i ≠ M+1 → a: (i) = 0; i := i+1 od
    ]|
    ; do n ≠ N
    →|[m, b: int; m, b := 0, 0
      ; do m ≠ M
        → |[c: int; c := a(m+1)
          ; if X(m) = Y(n) → a: (m+1) = 1 + b
            □ X(m) ≠ Y(n) → a: (m+1) = a(m) max a(m+1)
            fi
          ; b := c
          ]|
        ; m := m+1
        od
      ]|
    ; n := n+1
    od
    ; r := a(M)
    ]|
```

The computation time of this solution is $O(M \cdot N)$.

Solution of Exercise 29 (a search in Pascal's triangle)

We have to find a statement list S such that

$$
\begin{array}{l}
|[N\colon \mathit{int};\ \{N \geqslant 1\} \\
\quad |[k\colon \mathit{int}; \\
\qquad S \\
\qquad \left\{k = \left(\mathbf{MIN}x, y\colon 0 \leqslant x \leqslant y \wedge \binom{y}{x} = N\colon y\right)\right\} \\
\quad]| \\
]|
\end{array}
$$

Since $\binom{y}{x} = \binom{y}{y-x}$, we may restrict in the postcondition the domain of x to $0 \leqslant x \leqslant y$ **div** 2. Let

$$K = \left(\mathbf{MIN}x, y\colon 0 \leqslant x \leqslant y \ \mathbf{div}\ 2 \wedge \binom{y}{x} = N\colon y\right).$$

The postcondition is then $k = K$.

We adopt $P0 \wedge P1 \wedge P2$ as our invariant:

$$P0\colon \qquad 0 \leqslant j \leqslant k \ \mathbf{div}\ 2 \wedge \binom{k}{j} = n,$$

$$P1\colon \qquad \left(\mathbf{A}x, y\colon 0 \leqslant x \leqslant y \ \mathbf{div}\ 2 \wedge y < k\colon \binom{y}{x} \neq N\right).$$

Notice that $P1$ implies $k \leqslant K$, and that $P0 \wedge P1 \wedge n = N$ implies $k = K$.

$$P2\colon \qquad \left(\mathbf{A}x, y\colon j < x \leqslant y \ \mathbf{div}\ 2\colon \binom{y}{x} \neq N\right).$$

In the repetition either k is increased or j is decreased. First consider statement $k := k+1$. Obviously, it leaves $P2$ invariant. Since

$$0 \leqslant j \leqslant k \ \mathbf{div}\ 2 \wedge \binom{k}{j} < N$$

implies

$$\left(\mathbf{A}x\colon 0 \leqslant x \leqslant j\colon \binom{k}{x} < N\right)$$

and since $P2$ implies

$$\left(\mathbf{A}x\colon j < x \leqslant k \ \mathbf{div}\ 2\colon \binom{k}{x} \neq N\right),$$

we have

$$P0 \wedge P2 \wedge n < N \ \Rightarrow \ \left(\mathbf{A}x\colon 0 \leqslant x \leqslant k \ \mathbf{div}\ 2\colon \binom{k}{x} \neq N\right).$$

Consequently, if $n < N$ statement $k := k+1$ maintains invariant $P1$ as well.

Next consider statement $j := j-1$. It does not affect $P1$. Since $\binom{k}{j} > N$ implies

$$\left(\mathbf{A}y: y \geq k: \binom{y}{j} > N\right)$$

and since $P1$ implies

$$\left(\mathbf{A}y: j \leq y \,\mathbf{div}\, 2 \wedge y < k: \binom{y}{j} \neq N\right),$$

we have

$$P0 \wedge P1 \wedge n > N \;\Rightarrow\; \left(\mathbf{A}j: j \leq y \,\mathbf{div}\, 2: \binom{y}{j} \neq N\right).$$

Consequently, if $n > N$ statement $j := j-1$ maintains invariant $P2$ as well.

The way to maintain $P0$ under $k := k+1$ and $j := j-1$ follows immediately from

$$\binom{k+1}{j} = \left(\binom{k}{j} \cdot (k+1)\right) \Big/ (k+1-j)$$

and

$$\binom{k}{j-1} = \left(\binom{k}{j} \cdot j\right) \Big/ (k-j+1)$$

respectively.

Thus, our program will have the following structure.

$S0$ $\{P0 \wedge P1 \wedge P2\}$
; **do** $n < N \rightarrow k, n := k+1, (n * (k+1))/(k+1-j)$
□ $n > N \rightarrow j, n := j-1, (n * j)/(k-j+1)$
od$\{P0 \wedge P1 \wedge n = N$, hence $k = K\}$

The initializing statement $S0$ establishes

$$R: \quad k = \left(\mathbf{MIN}y: y \geq 0 \wedge \binom{y}{y \,\mathbf{div}\, 2} \geq N: y\right)$$
$$\wedge j = k \,\mathbf{div}\, 2 \wedge \binom{k}{j} = n.$$

Then $P0$ obviously holds. From the first conjunct of R follows

$$\left(\mathbf{A}y: 0 \leq y < k: \binom{y}{y \,\mathbf{div}\, 2} < N\right)$$

Let x and y be such that

$$0 \leq x \leq y \,\mathbf{div}\, 2 \wedge y < k.$$

Then R implies

$$\binom{y}{x} \leq \binom{y}{y \textbf{ div } 2} < N.$$

We conclude that R implies $P1$.

From R follows

$$\binom{k}{k \textbf{ div } 2} \geq N \wedge j = k \textbf{ div } 2$$

Let x and y be such that

$$k \textbf{ div } 2 < x \leq y \textbf{ div } 2.$$

Then R implies

$$\binom{y}{x} > \binom{y}{k \textbf{ div } 2} \geq \binom{k}{k \textbf{ div } 2} \geq N.$$

Consequently, R implies $P2$ as well.

We establish R by a linear search, maintaining

$$\binom{k}{k \textbf{ div } 2} = n.$$

The adjustment of n when increasing k by 1 follows from the following relations:

- for k even,

$$\binom{k+1}{(k+1) \textbf{ div } 2} = \left(\binom{k}{k \textbf{ div } 2} \cdot (2 \cdot k+2)\right) \Big/ (k+2),$$

- for k odd,

$$\binom{k+1}{(k+1) \textbf{ div } 2} = \binom{k}{k \textbf{ div } 2} \cdot 2.$$

We have now assembled all ingredients of our solution:

```
S:  |[n, j: int; k, n := 0, 1
    ; do n < N
      → if k mod 2 = 0 → n := (n * (2 * k+2))/(k+2)
        □ k mod 2 = 1 → n := n * 2
        fi
      ; k := k+1
      od
    ; j := k div 2
```

```
; do n < N → k, n := k+1, (n * (k+1))/(k+1−j)
  □ n > N → j, n := j−1, (n * j)/(k−j+1)
  od
]|
```

A suitable bound function for the second repetition is $K-k+j$, which shows (since $j \leqslant K$) the execution time of S to be proportioal to the row number computed.